Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1135 – 1148

doi:10.3906/elk-1907-13

Convolutional auto encoders for sentence representation generation

Ali Mert CEYLAN, Vecdi AYTAÇ*
Department of Computer Engineering, Faculty of Engineering, Ege University, Izmir, Turkey

Received: 02.07.2019 • **Accepted/Published Online:** 30.12.2019 • **Final Version:** 28.03.2020

Abstract: In this study, we have proposed an alternative approach for sentence modeling problem. The difficulty of the choice of answer, the semantically related questions and the lack of syntactic closeness of the answers give rise to the difficulty of selecting the answer. The deep learning field has recently achieved a pivotal success in semantic analysis, machine translation, and text summaries. The essence of this work, inspired by the human orthographic processing mechanism and using multiple convolution filters with pre-rendered 2-Dimension (2D) representations of sentences, input or output size is to learn the basic features of the language without concerns. For this reason, the semantic relations in the sentence structure are learned by the convolutional variational auto-encoders first, and then the question and answer spaces learned by the auto-encoders are linked with proposed intermediate models. We have benchmarked five variations of our proposed model, which is based on Variational Auto-Encoder with multiple latent spaces and able to achieve lower error rates than the baseline model, which is the base Convolutional LSTM.

Key words: Convolutional networks, bi-gram, n-gram, question answering problem, deep learning, variational auto-encoder, sentence modeling

1. Introduction

Sentence modeling problem is involved in almost all natural language processing problems. Neural network based models are shown to be quite advantageous compared to other models. They can be trained to predict words or phrases in a context or generate sentences word by word [1]. The relevance of an answer (theoretically) is determined by measuring the semantic similarity between question and answer. Prior work in the field of statistical Natural Language Processing mainly focused on this approach with the syntactic matching of parse trees. Also, there exist some studies that use WordNet to add semantic information to lexical representations [2]. Sentence modeling is usually involved with the extraction of parts of a sentence and it is related to identity, synonymy, antonymy, etc. However humans use the content of one sentence to provide the representation of another [3]. The answer selection problem can be described as a classification problem where a question corresponds to multiple ground-truth answers. Challenge of this problem comes with semantically related questions and answers that may not be lexically close. Deep learning models have recently obtained salient success on semantic analysis, machine translation, and text summarization [4]. With given word embedding's, vector representations of a sentence acquired by summing over the embedding's of all words in the sentence and then the vector is normalized by the length of the sentence which is known as the bag-of-words model. However, it loses word ordering during the process, which is the disadvantage of this model. Instead, bi-gram model does not loose word order. Furthermore, convolutional networks can learn deeper structural knowledge

*Correspondence: vecdi.aytac@ege.edu.tr

[2]. Even though bag-of-n-grams contains word order, it suffers from high dimensionality and sparsity [5]. Also, as it is mentioned in by [1], convolutional filters can learn n-gram representations less than or equal to their size, which is very important to determine irregular distributions of phrase-level structures in sentences. Some of the convolutional models use vector embeddings of words; [2, 4] with various techniques; [6, 7]. Also [2] reported that bi-gram model is observed to be achieving better performance with the help of Inverse Document Frequency (IDF) weighted word count features.

1.1. Neurological basis

From the physiological aspect, reading printed material is defined as "Orthographic Processing". This suggests that individual letter in a string of letters is processed for their identity and position. It is first proposed by Grainger and van Heuven by considering the independent processing of individual letters. They proposed detectors which are location specific -relative to eye fixation position- and reactive to certain visual features at a location, they named them as "alphabetic array".

However, this alphabetic array does not give the relative position of a letter since each of them is processed separately. On the other hand, letter identity is connected to its string position independently from its spatial position. Also, it has been mentioned in [8] that main idea of Grainger and van Heuven's study is sub-lexical orthographic representations coded in the lower levels of processing as continuous and noncontinuous letter sequences. Relative position coding of nonadjacent letters is also used in Whitney's SERIOL model. For example; CART coded as CA, CR, CT, AR, AT, RT [9]. In this example, nonadjacent letters also form a pair as a design consideration of the model, however by preserving letter order at the same time. This is called "Open bi-gram" coding by Grainger and van Heuven (2003), as mentioned in the [8].

This framework consists of several processing layers, which provide visual translation of visual input to the semantic bindings. The model has five layers: edge, feature, letter, bi-gram, and word. We will not cover the first three layers since the visual recognition of letters are not included within the scope of this study. In this model, bi-gram layer nodes are connected to the word layer with weighted connections. Thus, words are built upon the order of letters and sequences, but not exactly on string position. However, sequentiality comes from the nature of the reading behavior of humans. Therefore, in order to read a text, focusing on the letters sequentially triggers related bi-gram neurons consecutively.

Lower level feature extractors can only detect letters with a specific shape and they look for details such as lines at different angles, where the receptors have a small tolerance to the displacement of features. Abstract letter identities are detected at higher layers.

Even though smaller receptive fields limit the ability to detect changes in size and location, multiple smaller receptive fields placed in different locations create position-invariant local letter detectors. This described functionality is shown to be similar to what convolutional networks are used for.

For upper layers of visual hierarchy, visual receptors become progressively larger. Based on this assumption, a neurobiological scheme is designed where hierarchy becomes wider and more abstract neurons represent combinations of features that each being sensitive to local combinations of them. Increase in the size of receptive fields progressively triggers the neurons to code sequences of letters up to three letters. This trade-off between location invariance and selectivity neuron coding of a triplet is expected to occur less and more specific positions, where a neuron coding for a letter in three different positions has no meaningful information. Although such units may be contributing to feature extraction, bi-grams seem to be the best approach between location invariance and letter position coding.

Location invariance is not limited to a receptor field size of progressing layers but it can also be provided by pooling over partially crossing letter detectors. Thus, the broader receptive field will be used for processing information. Grainger and colleagues defined this type of coding as "open bi-grams" [8]. In this model, words are coded by a collection of pairs occurred in the text consecutively. The existence of such an architecture is not confirmed. However, the model helps to explain why neurons behave like bi-gram detectors.

Word layer also uses bi-gram layer features. Since bi-grams are ordered pairs of letters, it is natural to expect ordered pairs of bi-grams formatted words. As it is mentioned by the authors, words are never coded by a single neuron, instead by multiple lower level receptors.

It has been argued that this combinatorial scheme would cause an explosion of required detectors conversely increasing progressive layers receptive field and decreasing the number of receptors. However, tuning of lower level detectors is highly correlated with reading patterns and eye movements.

2. Design and implementation

Until now, several related topics are covered in their relevant sections. It can be noticed that neurological models have clean structure and component based similarities with the computational models proposed. As it was discussed before, Open bi-gram and SERIOL models depend on bi-grams to develop translation invariance while keeping word/letter order. Most of the neural models approach this issue as word bi-grams rather than letter bi-grams in the neurobiological model. It was considered as letter based because both models handle orthographic processing mechanism from recognition of letters to order of letter pairs, pairs of words, and lastly semantic relations they correspond to. Interestingly, computational models depend on translation invariance of convolution operation, and as it has been mentioned in the literature review, multiple stacked convolutional layers learn a hypothetical parse tree where activations of translation invariant features are pooled together to be processed by further convolutional layers, such as inter-word relations to phrase-word relations. Since we begin processing from the existing sentences, visual recognition part is not covered, but instead bi-grams are used as the base of our data representation. We have used Google's n-gram corpus from [10] for the bi-gram counts.

2.1. Data representation

In this study, we decided to use bi-gram frequencies of the words and letters based on the reasons stated in the previous chapter. Therefore, every sentence could be considered as a sequence of letters. Since bi-gram models can be considered as first-order Hidden Markov Models (HMMs), the underlying problem space could be considered as first-order arc-emission HMM. Hence, we based our data representations on top of the emission probabilities of consecutive letters. Each cell of the representation is calculated according to Algorithm 1.

The idea behind the transformation of a natural-language-processing problem to a visually interpreted problem is to make it possible to learn underlying features without explicitly pointing to them in the samples. In other words, we may consider it as the object recognition problem of computer vision, where the object is fully or partially exists in the image and our feature extractor somewhat detects certain properties of the partially visible object. Imagine that object is not even partially visible in the picture, but there exists certain background, environmental properties exist and indicate that such a tool must be the one related to this environment. For example, a saw handle can be barely visible in a toolbox and cannot be distinguished under isolation from the entire image it belongs. For this problem, the model aims to estimate the possible number of tools for specific environments by only seeing peripheral objects provided inside the images and has to make an

Algorithm 1: Bi-gram Transmission Frequency Matrix Creation Algorithm

Tokenized, cleaned list of words W
Processed bi-gram frequency count matrix S
$ps \leftarrow 0$ *start index of previous word*
$len_W \leftarrow$ length of W
$S \leftarrow$ regular grid with size $len_W x len_W$
for $t = 0$ *to* len_W **do**
    $L_W \leftarrow W_{t,t+2}$
    $len_{L_W} \leftarrow$ length of $W_{t,t+2}$
    for $i = 0$ *to* len_{L_W} **do**
        for $j = 0$ *to* len_{L_W} **do**
            $L_{freq_{i,j}} \leftarrow$ frequency count of $\log_{10}\left(L_{W_i} \& L_{W_j}\right)$
            if $S_{i+ps,j+ps} = 0$ **then**
                $S_{i+ps,j+ps} \leftarrow L_{freq_{i,j}}$
            else
                $S_{i+ps,j+ps} \leftarrow \frac{\log_{10}(S_{i+ps,j+ps}) + L_{freq_{i,j}}}{2}$
    $ps \leftarrow L_{W_0}$
    $\{S$ is resulting matrix of overlapped frequency counts of letters and words$\}$

estimate for this number by learning possible relations of objects that are indirectly related. That environment, background, related objects, and type of toolbox are now important properties to make an assumption on what that undistinguishable object might be. For our case, sentences are represented as a collection of vectors/word embeddings consecutively. The model learns from the order of the vectors/word embeddings, and probably internal properties of the embeddings. That case works because a speaking language is a human-made thing and its all aspects are well-defined, if it is wanted to be modeled or some heuristics wanted to be involved in any recognition model. However, in this study, we assumed that the language was unknown and only the frequency counts of symbols that were observed are observable by model. Thus, we used bi-gram frequencies of consecutive letters and words, and then used both of them in a 2D plane. The problem can also be treated with computer vision techniques, more importantly, models used can be learning relations between letters, words without explicitly seeing them or obtaining their feature representations. A model should learn to distinguish different letters by inter-letter properties so that it can have an internal representation for words, and then it can use it to learn inter-word relations, and so on.

2.2. Model

For the architecture proposed in this study, it is aimed to reduce number of parameters but to force the model to learn distinct features at the same time. We based our model over the Variational Auto-Encoder (VAE). Auto-Encoder (AE) is a model with a specially placed bottleneck in order to compress information in to a vector with pre-specified size. In order to achieve that network learns to encode input to a fixed size vector, and then reconstruct input from encoded vector. In this study, we have based our architecture over Convolutional Variational Auto-Encoder. In contrast to AE, VAE learns to encode in to two separate fixed size vectors, namely mean, μ and variance, $\log_{10}\left(\sigma^2\right)$. To produce vector/embedding, μ and $\log_{10}\left(\sigma^2\right)$ used to shift sample taken

from standard normal distribution $\mathcal{N}(0,1)$ (see Eq. 1).

$$z = \mu_i + e^{\log_{10}(\sigma_i^2)}\mathcal{N}_{(0,1)}. \tag{1}$$

Equation 1, actually called as *reparameterization trick* where $\boldsymbol{z}$ is a continuous random variable, and we use it for generating samples from conditional distribution $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|x)$. $q_\phi(\boldsymbol{z}|x)$ is the probabilistic encoder which will be an approximation of true posterior $p_\theta(\boldsymbol{z}|x)$, and we will call $p_\theta(x|\boldsymbol{z})$ as probabilistic decoder [11]. Recognition model parameters referred as ϕ and generative model parameters as θ. Marginal likelihood is the probability of an event occurring and which is the sum over the marginal likelihoods of individual data points.

$$p_\theta(x) = \sum_{i=1}^{N} \log_{10}\left(p_\theta\left(x^i\right)\right), \tag{2}$$

and $\log_{10}\left(p_\theta\left(x^i\right)\right)$ can be written as

$$\log_{10}\left(p_\theta\left(x^i\right)\right) = \mathcal{D}_{KL}\left(q_\phi\left(\boldsymbol{z}|x^{(i)}\right) \| p_\theta\left(\boldsymbol{z}|x^{(i)}\right)\right) + \mathcal{L}\left(\theta, \phi; x^{(i)}\right). \tag{3}$$

$\mathcal{L}(\theta, \phi; x)$ is the lower bound on the likelihood of datapoint i, and when $\mathcal{L}(\theta, \phi; x) = \log_{10}\left(p_\theta\left(x^i\right)\right)$, KL loss between two distribution, namely $q_\phi(\boldsymbol{z}|x)$ and $p_\theta(\boldsymbol{z}|x)$ will be zero therefore, likelihood $\mathcal{L}(\theta, \phi; x)$ will be maximized. For our purposes KL-divergence must be integrated in order to be implemented. A detailed proof can be found at Appendix B of [11]. Thus, it can be written as

$$\mathcal{D}_{KL}\left(q_\phi\left(\boldsymbol{z}|x^{(i)}\right) \| p_\theta\left(\boldsymbol{z}|x^{(i)}\right)\right) = 0.5 \sum_{j=1}^{N}\left(1 + \log_{10}\left(\sigma_j^{(i)}\right)^2 - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2\right). \tag{4}$$

Therefore, KL-loss is the sum of KL-divergences. For our case, we switch indices for symbolic representability and use j to refer a single VAE and i to an instance of the dataset

$$\mathcal{L}(\theta, \phi; x) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\mathcal{D}_{KL_j}\left(q_{\phi_j}\left(\boldsymbol{z}_j|x^{(i)}\right) \| p_{\theta_j}\left(\boldsymbol{z}_j|x^{(i)}\right)\right) + (x^{(i)} - \tilde{x}^{(i)})^2. \tag{5}$$

We have used standard VAE loss and considered Mean Squared Error (MSE) as reconstruction loss (see Figure 1). N stands for the number of records in our dataset, and M stands for the number of VAE used in the model. For the special case of VAE loss function includes a term to measure how much re-generated inputs are diverging from the original inputs. The KL-divergence term can then be interpreted as regularizing encoder parameters, encouraging the approximations to be close to the original data distribution [11]. We aimed to reduce the number of the parameter but to force model to learn distinct features at the same time. To do that, we dedicated separate smaller scale VAE for each channel of the output of prior convolutional layer. Feature map of each channel is processed separately in the VAEs and reconstructions are concatenated and fed into the transposed convolutional layer. This way we overcome parameter explosion in the previous architecture described.

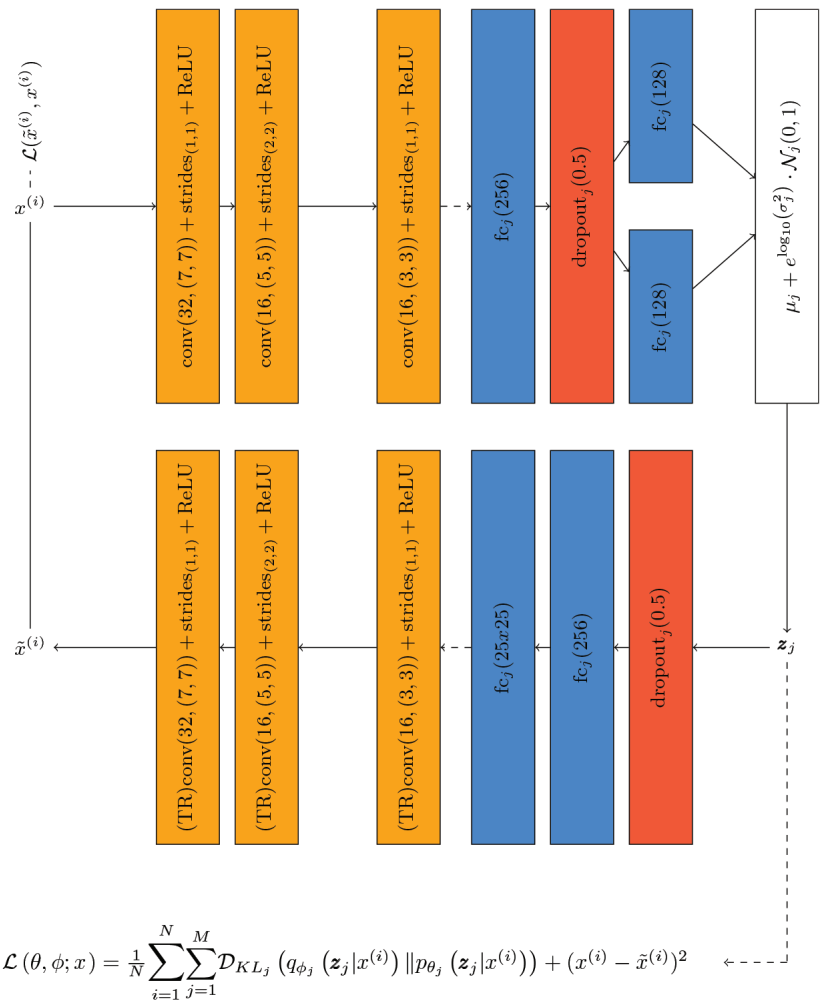

Figure 1. CMCAVAE, convolutional architecture that consists of two consecutive convolutional layers accompanied by transposed convolutions. TR refers to Transpose of convolution operation.

2.2.1. Design

We focused on to design a model that uses the least number of filters with larger size in order to learn general scheme of syntactic representations buried inside transmission probabilities of letters and words. For this reason, we have designed and used two convolutional layers for encoder and decoder. In order to provide some translation invariance, second convolutional layers strides are set to two. Allowing the model to learn inter-feature dependencies itself seems more reasonable rather than letting it figure out operation scheme of the max-pooling operator, as mentioned in [12]. However, during internal comparisons, we concluded no difference of convergence between both applications due to the architecture in this study. This may be caused by higher equivariance by of using fewer convolutional layers compared to deeper convolutional networks. For the decoder part same number of layers applied with the same stride settings at corresponding symmetrical opponents of encoder layers. Even though [8] suggests, the increased size of receptor fields at deeper levels of detection, we found no difference was observed between neither convergence nor lowest error point of two approaches in this study (see Figure 2).

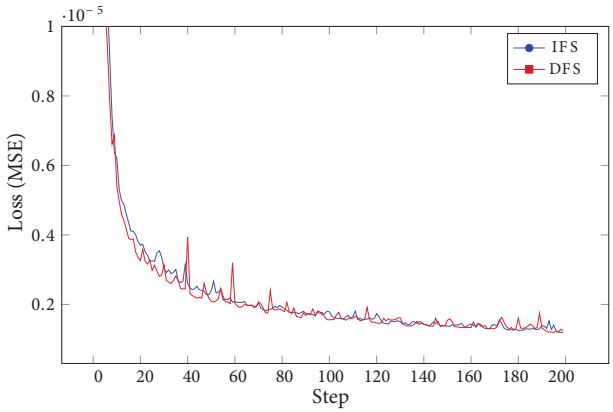

Figure 2. Increasing and decreasing filter size validation losses.

VAE learns to map the distribution of the data that is reconstructed to a n-dimensional random normal distribution. Since stacking multiple layers causes a parameter explosion on the forthcoming layers, single VAE is separated to multiple parallel ones that each aligned to a channel of the convolutional layer (see Figure 1). Channel aligned part of the model begins with a convolutional layer, which has 16 filters and each channel of convolution output is directed to a single VAE. Thus, we have $M = 16$ VAE where each channel of the convolution layer output mapped to a VAE. We have called this specific type of architecture as Convolutional Multiple Channel Aligned Variational Auto-Encoder (CMCAVAE). This scheme is expected to increase discrimination of detected features among the filters. Since convolutional layers are expected to replace parse trees, this type of alignment would cause higher distinction of features by enforcing separate error back-propagation to their own specific kernel. This type of alignment of VAE is inspired from biological equivalents where each neuron has multiple synapses to post-synaptic cells and activation potential depends on "The amount of transmitter released is determined by the number and frequency of the action potentials that reach the presynaptic terminals" [13, p. 35]. This type of activation involves at least two external parameters received by the cell. If we consider synaptic spaces -where neurotransmitter matter travels to post-synaptic cell-, there is also re-absorption of neurotransmitter released from pre-synaptic cell to be considered as an environmental property that affects activation potential at a post-synaptic cell. In addition, one has to consider that post-synaptic ion channels might be addressed to a certain type of neurotransmitter.

Inspiring from synaptic structures, we considered latent space of VAE as synaptic space hence we decided to use multiple variational auto-encoders at a smaller scale, by this way, parameter explosion is prevented and our multiple variational auto-encoder models outperform single VAE model which has parameters higher 16 times. Although that the model had more parameters, it was observed to report NaN loss. Even though some regularization and gradient clipping were applied, it had only delayed the fall out of the convergence.

Using multiple variational auto-encoders and forcing each of them to learn from feature map of a single convolution kernel might help them to map separate parts of distribution to different latent spaces.

We used a connection dropout rate with 0.5 probability as it is advised by [14] to prevent VAEs from overfitting so that they can handle recognition task evenly. For our case, effect of dropout can be observed very clearly in Figure 3. Dropout is a method of regularization where either specified percentage of randomly selected neurons from a layer masked, so output connections will not be added to activations of next layer, or specified percentage of randomly selected weights are masked. During back-propagation, these masked neurons or connections are not updated according to back-propagation rules.

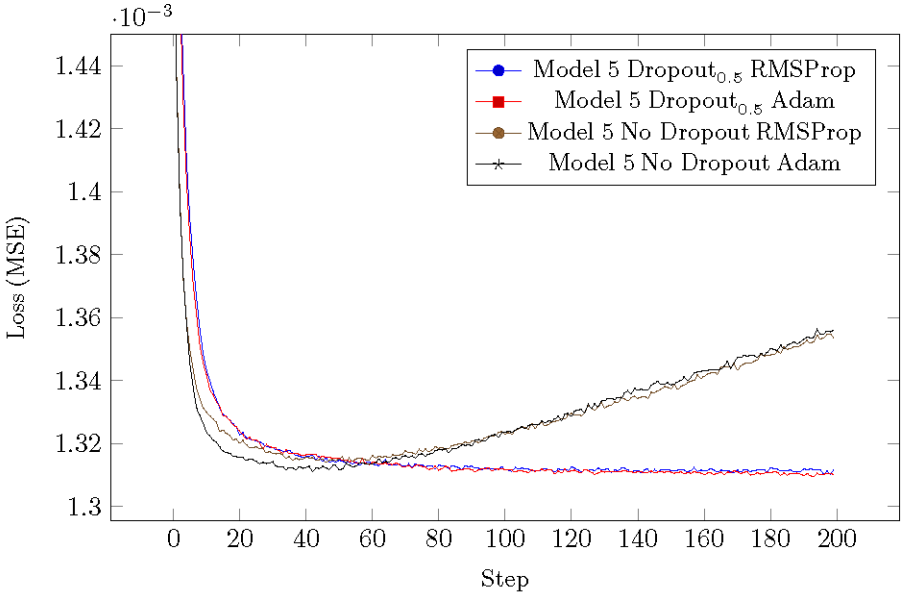

Figure 3. Validation loss of model 5 with Adam/RMSProp and with/without dropout. It is very clear that the model overfits without dropout.

Convolutional layer is used before VAEs aims to hold the position of a separation point where each filter extracts feature for different latent space encoded.

2.2.2. General model for sentence representation generation

In this subsection, we propose several alternative models and their comparison with the baseline on learning to generate sentence representations that match the ground truth answer. The objective of this test is to compare convergence capabilities and performance of various architectures of various parameter numbers. Most of the models shown in Figure 4 have an approximately same number of parameters and some of them have significantly fewer parameters, but reach comparable learning performance on this specific measurement. Additionally, one can check the visual representation of an answer from the dataset, as well as answers generated by one of our proposed model and baseline model in Figure 1. Traning a full Question Answering (QA) model consists of multiple stages as follows:

1. Training a multiple CMCAVAE for the question and answer dataset separately.
2. Using encoder of question CMCAVAE and decoder of answer CMCAVAE, training intermediary model which is capable of re-creating language features, and an approximation of answers.
3. Model itself can be trained specifically for answer selection task.

All the intermediary models proposed are receiving mean and variance vectors produced by VAEs. Mean and variance vectors which it has output are sampled by sampling layers of decoder VAEs.

3. Experiments

During the experiments with the various intermediate and latent dimension configurations, -among the models we have proposed- we have observed that latent dimension is more important than a large number of convolutional layers when it comes to the learning distribution. For this and computational reasons the number of VAE

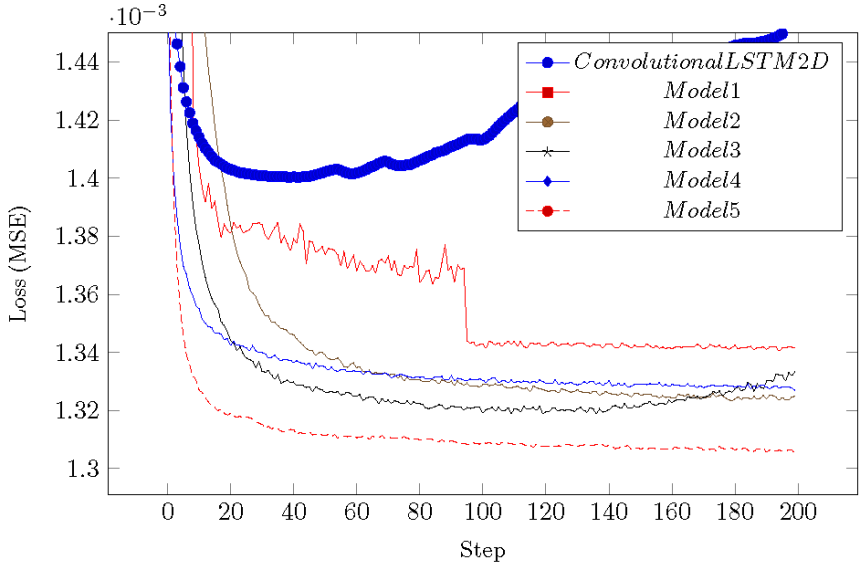

Figure 4. Convergence plot of multiple architectures under same conditions.

is limited to 16 and intermediate latent dimension sizes are increased. For the baseline, we considered Convolutional Long Short-Term Memory (LSTM) model that is proposed in [15]. Convolutional LSTM originates from LSTM [16] is widely used for learning sequential data, and natural language processing. However, for convolutional LSTM, kernels have their gating mechanism and this provides convolutional architecture to have control over its own output. LSTM is a variation of Recurrent Neural Network (RNN). It has self-connected units that allow information flow or prevent it. In contrast to RNN it has an input and output gate units in order to protect the cell from irrelevant inputs or other cells from the irrelevant output of the cell. More specifically it has a forget gate, state, external input gate and output gate units.

Convolutional LSTM has both sparse interactions with the input and hidden states to temporal features. However, it was unable to establish a stable convergence curve with its current configuration. We have preferred a mini-batch size of 32 for all models compared in this study. We have experimented with several other mini-batch sizes, and we have observed good results in validation with larger ones even though it is not advised to use sizes larger than 32. Dataset consists of first 32000 dialog pairs that the sentence length is less than 50 characters of Cornell Movie Dialog Corpus [17], and it has been split by the ratio of 0.8 and 0.2 for train and validation, respectively. All the plots shown in this study are the validation losses of models. The separate test results are not included due to the fact that even the baseline model has failed to reach a reasonable reconstruction loss. Since the purpose of these tests is the measuring capacity of generative models for this data representation in English language, model states are reset at each instance for recurrent models, thus they will not be able to learn the data distribution in the dataset. Glorot Uniform [18] left as default weight initializer method for all convolutional, dense and LSTM layers where $U\left[-a,a\right]$ denotes to uniform distribution and n is the size of previous layer in the network.

$$W_{i,j} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \tag{6}$$

Activation functions used for LSTM layers are left as hyperbolic tangent. It covers larger output value range than logistic sigmoid and this extended domain reduces saturation at the edges of the function. Logistic

sigmoid function is producing saturating values close to the edges of its domain. [19] says, hyperbolic tangent function typically performs better than logistic sigmoid and refers to $tanh(0) = 0$, which tangent activation function similar to identity function near range of 0. This makes tangent function a better candidate for fitting linear models.

As optimizer, we mainly stick to Adaptive Moment Estimation (Adam) [20], with default parameters. Adam optimizer only uses directions of the first order gradients, not the values of the gradients. This way it can control weight update with a moving average of previous k gradients at the direction of the gradient calculated the error at time step t. Additionally Adam has self-regularization mechanism to prevent initial biased average of gradients to effect direction of convergence. This additional process is called as bias-correction.

For Adam, m_0, v_0 are 1st and 2nd moment vectors of gradients, θ_t is parameter at time step t, α is the learning rate and $\epsilon = 10e - 8$

$$g_t = \frac{\partial \mathcal{L}\left(\theta, \phi; x\right)}{\partial \theta_{t-1}}, \tag{7}$$

First moment estimate

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{8}$$

Second moment estimate

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{9}$$

So, parameter update is achieved according to

$$\theta_t = \theta_{t-1} - \alpha \frac{\frac{m_t}{(1-\beta_1^t)}}{(\sqrt{\frac{v_t}{(1-\beta_2^t)}} + \epsilon)}. \tag{10}$$

On the other side, "Root Mean Square prop (RMSprop) with momentum generates its parameter updates using a momentum on the rescaled gradient, whereas Adam updates are directly estimated using a running average of first and second moment of the gradient" [20].

3.1. Model 1

Model 1 consists of multiple fully connected layers (see Figure 1) where an information bottleneck is placed on purpose, in order to force the model to make generalization. Before the second hidden layer, a connection dropout layer with 0.5 probability is placed. The inspiration for this type of installment comes from denoising auto-encoders where input data points are corrupted by Gaussian noise, and the model is expected to produce output that is complete/recovered version of the input. This learning strategy is expected to capture real data distribution rather than the distribution of input samples used in the training dataset. With these insights, the model has achieved fairly well convergence with some unexpected improvements. The dramatic drop in the loss, which is shown in Figure 4, and followed by plain, stable with minor variations up until another drop in the next 200 iterations. However, it was not that much of decrease and it was not included in comparative analysis.

3.2. Model 2

Model 2 solely consists of 1D convolutional layers. The motivation behind it comes from the unprecedented success of 2D convolution operation on visual processes. Vectors derived from each variational auto-encoder does

Layer Configurations	
Model 1	32-7x7-16-5x5-16-3x3-z-16-3x3-16-5x5-32-7x7
Model 2	32-5x5-16-7x7-16-3x3-z-16-3x3-16-7x7-32-5x5

Table 1. Model 1 with decreasing filter sizes (DFS), and Model 2 with increasing filter sizes (IFS).

not depend on each other. However, they may unconditionally be inclined to detect similar varying features. Information encoded in the vectors -which are concatenated $mean$ and $log - variance$ of each variational auto-encoder- encodes to a specific point in the normal random distribution, and it is more important to learn which variational auto-encoder produced it rather than where a specific feature repeated. For this reason, 1D convolution has to be equipped with some form of spatial reduction technique. We considered max-pooling operation between the stack of 1D convolutional layers. Towards to end of the model, two upsampling layers used to enlarge, compressed input.

At the end of the intermediary model, single fully-connected layer applied to each vector, this procedure is called Time Distributed (TD) where each 32 vector produced considered if they were some sequentially dependent vectors -they are not- and single fully-connected layer applied to each vector independently. Even though our early experiments show that increasing/decreasing filter sizes do not affect convergence directly (see Table 1), filter sizes are set in decreasing order to force model to capture neighbor embedding patterns as it passes through subsequent layers (see Figure 2). Since those layers have smaller receptive fields they are forced to learn closer features occur between vectors. However, 1D convolution is observed to work best with LSTMs, which will be discussed later.

3.3. Model 3

Model 3 is designed to be the successor of model 2, the number of filters at convolutional layers are increased to 64. However, the increase in the number of parameters without regularization caused overfitting in first 200 iterations. However, when compared to many stacked fully connected layers and LSTMs, models 2 and 3 have significantly fewer parameters, and having observed overfitting raises questions about architectural issues in these models. The reason behind this conclusion is that the sparse interaction of 1D convolution is only limited to patterns occur between consecutive vectors, not the ones occur among the vector on itself, so translation invariance provided by this architecture is limited by itself. We have also observed that models 1, 4, and 5 have stacked fully connected layers and LSTMs and they follow a more stable convergence in time.

3.4. Model 4

In model 4, it is aimed to cover weak points and merge fully connected layers with the LSTM. The first layer of it is a fully connected layer where the number of parameters is $16x128$, as a reduction of the number of parameters by half (see Figure 3). It is expected to learn relations between mean and variance vectors of each VAE, so that LSTM can be fed with data that are more pre-processed. However, such a pre-processing step limits LSTMs ability by its own ability to discriminate important features between mean and variance vectors and vectors of other VAEs. LSTM is fed with a vector shape of $(16, 128)$ and has 32 units so that it has the higher capacity - the number of parameters- and might memorize directly. However, memorization of input distribution is not directly possible due to the pre-processing layer before LSTM. After LSTM, we use a fully connected layer with $(32x128)$ units before feeding it into the answer decoder.

In model 5, it was decided to replace first fully connected layer with a pooling layer because of the limitations we described previously. For this replacement, we decided to transpose $32x128$ feature map and used max-pooling afterwards. This type of transformation acts as a filter for activations on each individual vector rather than pooling between vectors. The remaining part of the architecture is denoted in same as the Figure 4.

4. Conclusion and future work

Considering our data representation, Convolutional LSTM is expected to perform better against the models we have proposed. However, it is reasonable to think that visualized features are not the ones we want the model to learn, since that Convolutional LSTM is unable to generalize that information and instead starts memorizing the train data. In other words, this data representation does not directly represent data points (letters and words for this case) but instead, it shows relative probabilities of passing one state to another, one letter to another and one word to another. This makes it a more difficult problem since the features to be learned are impossible to be observed and must be internalized by the model. This could be the reason why our models has followed a lower error rates with more stable convergence compared to the direct approach with Convolutional LSTM.

Dimensionality reduction applied in the architectures of model 4 and 5 are considered necessary in the future models. Other than mean and variance of the encoded distribution, it might require some additional specific evaluation in order to create more meaningful activation. Thus, we added dimensionality reduction to $16x128$ as a baseline to follow.

In model 4, a fully connected layer is used to reduce the shape of feature map, and therefore it prevents LSTM from memorizing and directly learning from the real distribution. However, it also distorts real data distribution and may be accompanied by a batch normalization layer afterwards.

In model 5, Tanh in LSTM may be replaced with another non-linear activation function because neither encoder nor decoder is trained to handle activations below 0. Fully connected layer with a Rectified Linear Unit (ReLU) activation is used to clear up them however, this squeezes the model in to an activation range where activation function itself not designed to be, for this reason, it can be replaced in the future. An alternative for decoder activation function can be considered as LeakyReLU, which is used in Generative Adversarial Networks to keep gradients from getting sparse by allowing small negative values to pass [21]. 1D-pooling after transpose layer seems to be working better than the fully connected layer in Model 4. Also, the latest results show that, -see Figure 4- this method can be developed in order to improve performance.

MSE was used to measure reconstruction loss in the all models. However, it is not a convenient measure for such a deep task buried inside a simple data representation and it does not seem to discriminate between reconstructed parts and cannot relate to the length of output.

This type of data representation and learning may enable possibilities to learn unknown languages or sequential patterns, such as Deoxyribo Nucleic Acid (DNA). HMMs are mostly used in the area field of Bioinformatics for sequence comparison -aligning a sequence against a profile-, locating and defining suppressed parts of a gene (which is called as CG islands) [22]. These problems are important because aligning multiple DNA sequences raises computationally exhaustive challenges. CG islands, on the other hand, has importance since they define transcription start sites. HMMs are also used for unknown parameter estimation, however for our case, when we consider nucleotides as states which emitted with emission probability of 1.0, we have to have transmission probabilities between nucleotides in order to convert the string into the representation of transmission probabilities.

For this reason, DNA and Ribo Nucleic Acid (RNA) chains can directly be represented as it has been described in this study. This could allow us to extract meaningful features or subroutines by combining/customizing proposed models to have unsupervised feature extraction. With existing neural models, it might even be possible to create structural hierarchy encoded in a DNA. Besides the best assumption, even for the worst case, size-invariant running capabilities of convolutional networks could allow us to process very long DNA sequences all together at once. When multiple DNA sequences need to be aligned, they can be processed faster with a multi-channel convolutional network. However, as volumetric convolution mostly used in the field of medical imaging and since [23] proposed that 3D convolution produces larger feature-maps, it might require more training data in order to generalize matching parts of representation and extract more robust features to be used for profile alignment. For such a task, candidate model must be trained on varying length and number of DNA representations in order to generalize such a capability successfully. Even though overfitting problem can be solved by some structural and computational constraints of regularization, it might not promise a success unless sufficient amount of training data is available.

However, as a downside of neural language models, [24] suggests that performance of a basic encoder-decoder deteriorates rapidly as the length of an input sentence increases. This actually can be considered as fundamental design mistake which cannot be easily replaced. Neural network models, including some convolutional ones, are based on encoder-decoder architectures and they are trained end-to-end fashion. This forces the model to encode everything, however, natural languages have a sequential and progressive structure. The alignment and translation mechanisms proposed in the [24] could allow usage of architectures, (developed in this study) for profile alignment task more intuitively by allowing the model to correct itself while aligning multiple sequences.

References

[1] Kalchbrenner N, Grefenstette, E Blunsom P. A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics; Baltimore, Maryland, USA; 2014. pp. 655-665. doi: 10.3115/v1/P14-1062

[2] Yu L, Hermann K M, Blunsom P, Pulman S. Deep learning for answer sentence selection. arXiv preprint 2014; arXiv:1412.1632.

[3] Yin W, Schütze H, Xiang B, Zhou B. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. Transactions of the Association for Computational Linguistics 2016; 4: 259-272. doi: 10.1162/tacl_a_00097

[4] Tan M, Santos Cd, Xiang B, Zhou B. Lstm-based deep learning models for non-factoid answer selection. arXiv preprint 2016; arXiv:1511.04108.

[5] Le Q, Mikolov T. Distributed representations of sentences and documents. In: International Conference on Machine Learning; Lake Beijing, China; 2014. pp. 1188-1196.

[6] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning; Helsinki, Finland; 2008. pp. 160-167.

[7] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems; Lake Tahoe, Nevada, 2013. pp. 3111-3119.

[8] Cornelissen P, Hansen P, Kringelbach M, Pugh K. The Neural Basis of Reading. China: Oxford University Press, 2010.

[9] Whitney C. How the brain encodes the order of letters in a printed word: The seriol model and selective literature review. Psychonomic Bulletin & Review 2001; 8 (2): 221-243. doi: 10.3758/BF03196158

[10] Michel JB, Shen YK, Aiden AP, Veres A, Gray MK et al. Quantitative analysis of culture using millions of digitized books. Science 2011; 331 (6014): 176-182. doi: 10.1126/science.1199644

[11] Kingma DP, Welling M. Auto-encoding variational bayes. arXiv preprint 2013; arXiv:1312.6114.

[12] Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M. Striving for simplicity: The all convolutional net. arXiv preprint 2015; arXiv:1412.6806.

[13] Kandel ER, Schwartz JH, Jessell TM, Siegelbaum SA, Hudspeth AJ et al. Principles of Neural Science. New York, USA: McGraw-hill, 2000.

[14] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 2014; 15 (1): 1929-1958.

[15] Xingjian S, Chen Z, Wang H, Yeung DY, Wong WK et al. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems; Montreal, Canada; 2015. pp. 802-810.

[16] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation 1997; 9 (8): 1735-1780. doi: 10.1162/neco.1997.9.8.1735

[17] Danescu-Niculescu-Mizil C, Lee L. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In: Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics; Stroudsburg, PA, USA; 2011. pp. 76-87.

[18] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS); Chia Laguna Resort, Sardinia, Italy; 2010. pp. 249-256.

[19] Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge, Mass., USA: MIT Press, 2016.

[20] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv preprint 2014; arXiv:1412.6980v9.

[21] Chollet F. Deep Learning with Python. USA: Manning Publications Co., 2017.

[22] Jones NC, Pevzner PA. An Introduction to Bioinformatics Algorithms. USA: MIT Press, 2004.

[23] Lu L, Zheng Y, Carneiro G, Yang L. Deep Learning and Convolutional Neural Networks for Medical Image Computing. Switzerland: Springer, 2017, pp. 35-48.

[24] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint 2014; arXiv:1409.0473.

A. Data representation

a) Original b) Convolutional LSTM c) Model 5

Figure A 1. Original representation of the answer, the one created by the baseline and the one created by model 5. Re-creations are generated according to the same specific question in the dataset after 200 epoch.

B. Models

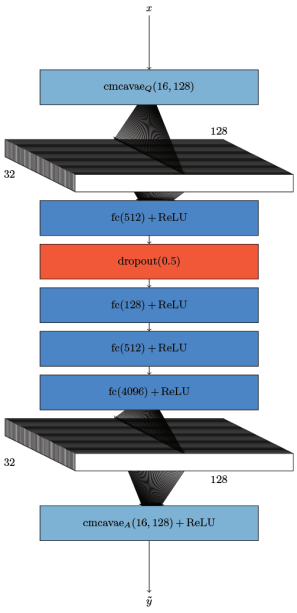

Figure B 1. Model 1

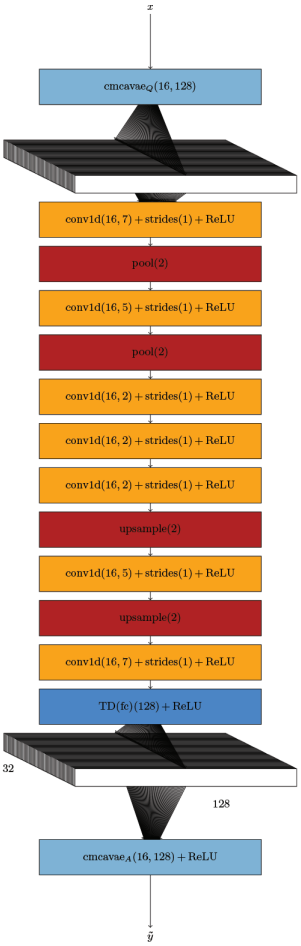

Figure B 2. Model 2, TD refers to Time Distributed, where same layer applied to a time series.

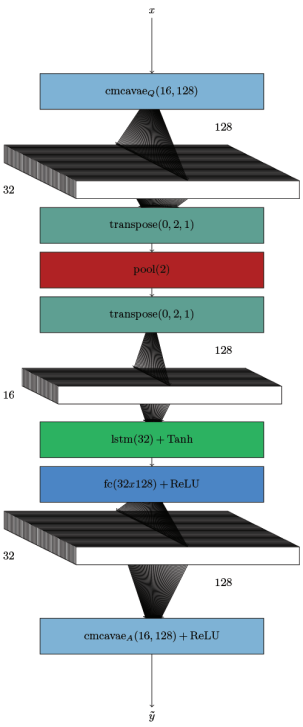

Figure B 3. Model 4

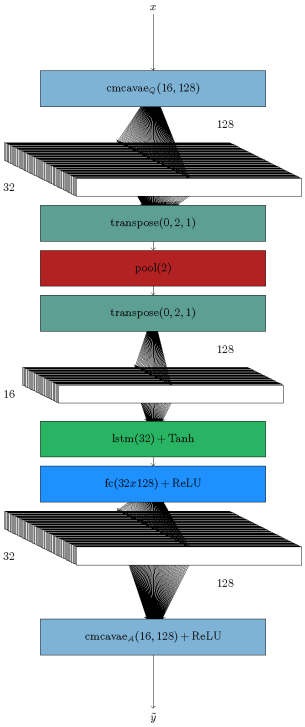

Figure B 4. Model 5